ROYAL INSTITUTE
OF TECHNOLOGY

ID2203 - Distributed Systems, Advanced Course

Exam Preparation

Course leader: Professor Seif Haridi

Assistant: Lars Kroll

{haridi, lkroll}@kth.se

Exam Structure

- 2 Sections, 50P total + quizzes, programming exercises, and project
    - 30P Multiple Choice Questions
    - 20P Reasoning Questions
- 4h total time
- “Closed book”
    - May take a dictionary, if you need it

ROYAL INSTITUTE OF TECHNOLOGY

MCQ Example

Let $v(e)$ denote the vector clock of event e, and $t(e)$ denote the Lamport logical clock of event e. Which of the following statements are true?

(a) $v(a) < v(b)$ implies that $t(a) < t(b)$

(b) $t(a) < t(b)$ implies that $v(a) < v(b)$

(c) $v(a) < v(b)$ implies that $\neg(t(b) < t(a))$

(d) $t(a) < t(b)$ implies that $v(a) \leq v(b)$

KTH VETENSKAP OCH KONST

ROYAL INSTITUTE OF TECHNOLOGY

MCQ Example

Let $v(e)$ denote the vector clock of event e, and $t(e)$ denote the Lamport logical clock of event e. Which of the following statements are true?

(a) $v(a) < v(b)$ implies that $t(a) < t(b)$

(b) $t(a) < t(b)$ implies that $v(a) < v(b)$

(c) $v(a) < v(b)$ implies that $\neg(t(b) < t(a))$

(d) $t(a) < t(b)$ implies that $v(a) \leq v(b)$

MCQ Example

Let $v(e)$ denote the vector clock of event e, and $t(e)$ denote the Lamport logical clock of event e. Which of the following statements are true?

(a) $v(a) < v(b)$ implies that $t(a) < t(b)$

(b) $t(a) < t(b)$ implies that $v(a) < v(b)$

(c) $v(a) < v(b)$ implies that $\neg(t(b) < t(a))$

(d) $t(a) < t(b)$ implies that $v(a) \leq v(b)$

ROYAL INSTITUTE OF TECHNOLOGY

MCQ Example

Let $v(e)$ denote the vector clock of event e, and $t(e)$ denote the Lamport logical clock of event e. Which of the following statements are true?

(a) $v(a) < v(b)$ implies that $t(a) < t(b)$ +1/2P

(b) $t(a) < t(b)$ implies that $v(a) < v(b)$ -1/2P

(c) $v(a) < v(b)$ implies that $\neg(t(b) < t(a))$ -1/2P

(d) $t(a) < t(b)$ implies that $v(a) \leq v(b)$ +1/2P

MCQ Point Total

- The MCQ part of the exam is subdivided into multiple subsections (e.g., *Basic Abstractions & Failure Detector*)
- Each section has an associated point total (2P/Question)
- Section point total with will be `max(0, s)`, where `s` is simply the sum of all individual questions within the section
- This means negative points do not carry across sections!

What to learn?

- All of the formal definitions
- All of the system/failure models
- All of the abstractions (their properties)
- Relationships between the abstractions (reductions)
- The high level mechanisms that make the algorithms work, e.g.
  - Read-Impose mechanism
  - Paxos invariants
  - log reconciliation in Raft

ROYAL INSTITUTE OF TECHNOLOGY

What not to learn?

- Correctness Proofs for the algorithms
  - Though it might help you learn the mechanisms to read them again
- Pseudocode for the algorithms
  - You'll be given that in the exam, if required

Exercise 1

Does the following statement satisfy the synchronous-computation assumption?

On my server, no request ever takes more than 1 week to be processed.

ROYAL INSTITUTE OF TECHNOLOGY

Exercise 1

Does the following statement satisfy the synchronous-computation assumption?

On my server, no request ever takes more than 1 week to be processed.

Yes! Known constant bound: 1 week

Exercise 2

Can we implement the perfect failure-detector abstraction in a model where the processes may commit omission faults and where we cannot bound the number of such faults?

What if this number is bounded but unknown?

What if processes that can commit omission faults commit a limited and known number of such faults and then crash?

Exercise 2

Can we implement the perfect failure-detector abstraction in a model where the processes may commit omission faults and where we cannot bound the number of such faults?

Definitely, not! Whatever timeout we pick, we can violate the *strong accuracy* property.

What if this number is bounded but unknown?

What if processes that can commit omission faults commit a limited and known number of such faults and then crash?

Exercise 2

Can we implement the perfect failure-detector abstraction in a model where the processes may commit omission faults and where we cannot bound the number of such faults?

Definitely, not! Whatever timeout we pick, we can violate the *strong accuracy* property.

What if this number is bounded but unknown?

Still, no, but: We could do EPFD at least!

What if processes that can commit omission faults commit a limited and known number of such faults and then crash?

Exercise 2

Can we implement the perfect failure-detector abstraction in a model where the processes may commit omission faults and where we cannot bound the number of such faults?

Definitely, not! Whatever timeout we pick, we can violate the *strong accuracy* property.

What if this number is bounded but unknown?

Still, no, but: We could do EPFD at least!

What if processes that can commit omission faults commit a limited and known number of such faults and then crash?

Actually, yes.

ROYAL INSTITUTE OF TECHNOLOGY

Exercise 3

In a fail-stop model, mark the following properties as safety or liveness.

L 1. every process that crashes is eventually detected

S 2. no process is detected before it crashes

S 3. no two processes decide differently

S 4. no two correct processes decide differently

S 5. every correct process decides before `t` time units

L 6. if some correct process decides then every correct process decides.

Exercise 4a

Suppose an algorithm A *implements a distributed programming abstraction* M *using a failure detector* D *that is assumed to be eventually perfect. Can* A *violate a safety property of* M *if* D *is not eventually perfect, for example, when* D *permanently outputs the empty set?*

KTH VETENSKAP OCH KONST

ROYAL INSTITUTE OF TECHNOLOGY

Exercise 4a

Suppose an algorithm A *implements a distributed programming abstraction* M *using a failure detector* D *that is assumed to be eventually perfect. Can* A *violate a safety property of* M *if* D *is not eventually perfect, for example, when* D *permanently outputs the empty set?*

No, because if that were possible A could already violate the same property if D were eventually perfect (during the non-perfect time)

Exercise 4b

Suppose an algorithm A *implements a distributed programming abstraction* M *using a failure detector* D *that is assumed to be eventually perfect. Can* A *violate a safety property of* M *if* D *is not eventually perfect, for example, when* D *permanently outputs the empty set?*

Now, what about a liveness property?

ROYAL INSTITUTE OF TECHNOLOGY

Exercise 4b

Suppose an algorithm A *implements a distributed programming abstraction* M *using a failure detector* D *that is assumed to be eventually perfect. Can* A *violate a safety property of* M *if* D *is not eventually perfect, for example, when* D *permanently outputs the empty set?*

Now, what about a liveness property?

Sure, anything that relies on at least some nodes being alive to make progress can be violated like this.

ROYAL INSTITUTE OF TECHNOLOGY

Exercise 5

Can we devise a broadcast algorithm that does not ensure the causal delivery property CB *but only its nonuniform variant* $\mathrm{CB}_{\mathrm{NU}}$*?*

$\mathrm{CB}_{\mathrm{NU}}$*: "No correct process p delivers a message* m_2 *unless p has already delivered every message* m_1 *such that* $m_1 \rightarrow_H m_2$*."*

Note: Processes may not self-destruct ;)

Exercise 5

Can we devise a broadcast algorithm that does not ensure the causal delivery property CB *but only its nonuniform variant* CB_{NU}*?*

CB_{NU}: *"No correct process p delivers a message* m_2 *unless p has already delivered every message* m_1 *such that* $m_1 \rightarrow_H m_2$*."*

No, because it automatically fulfils (uniform) *causal delivery*.